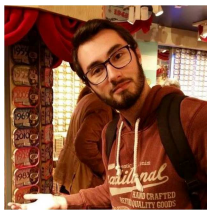
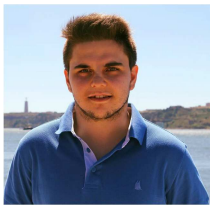
Sistemas Distribuídos 2016/2017

4ª Parte - Relatório de Tolerância a Faltas

Professor Naércio Magaia

Turno de 2ª feira, às 11h

Grupo A45 - https://github.com/tecnico-distsys/A45-Komparator

Rafael Belchior 80970	João Trindade 80805	Pedro Gomes, 81534
		

Lisboa, 19 de Maio de 2017

A nossa política de tolerância a faltas

O nosso sistema é assíncrono, pois não há garantias que as mensagens enviadas chegam ao destino dentro de um tempo limite. Também não há garantias que o tempo de execução de determinada operação está entre limites específicos.
Pretendemos tolerar faltas temporárias, através da semântica no máximo uma vez. Estas faltas temporárias podem ser, por exemplo, uma falta de energia que leva a que o mediador primário vá abaixo.
Tentamos maximizar a disponibilidade do nosso sistema ao criar um mediador secundário/backup, que fica responsável pelo processamento dos pedidos dos clientes, caso o mediador principal vá abaixo.
A política de tolerância a faltas baseia-se num mecanismo redundante, que permite que a função da componente comprometida seja obtida de outra forma.

Temos redundância na forma **física**, obtida através da duplicação de informação. Esta duplicação é conseguida através dos métodos updateShopHistory() e updateCart(), e através da substituição do servidor, caso este fique inacessível.

A redundância na forma **temporal** é obtida através da implementação do Front-End. Este permite repetir um pedido caso haja algum problema.

A redundância da **informação** é obtida através da implementação do handler MessageIdHandler. Este implementa a semântica no máximo uma vez.

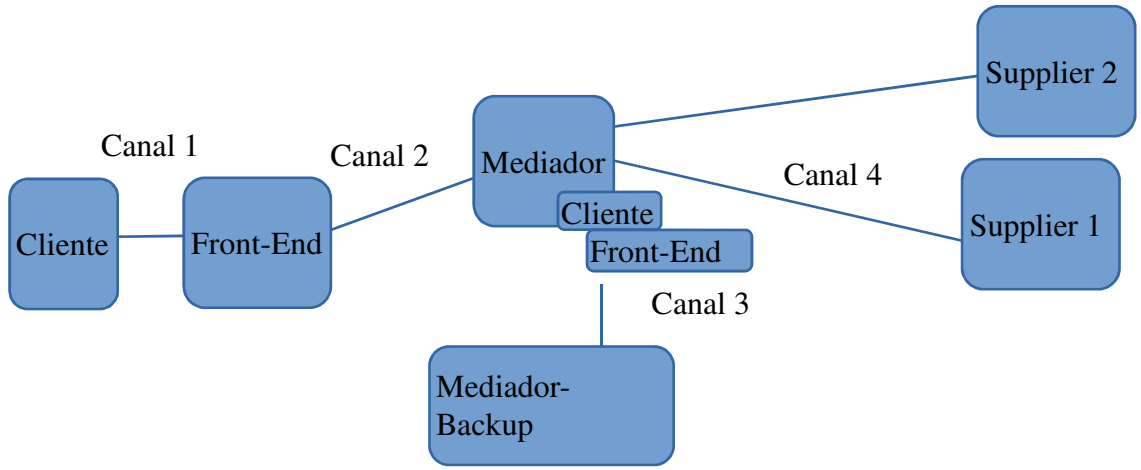


Figura 1- Esquema simplificado da solução de tolerância de faltas

Nota: A comunicação entre o mediador e o mediador backup é feita através de um mediador cliente. Este cliente, por sua vez, cria um front-end, que vai efetuar a comunicação (lidando com eventuais falhas).
Esta comunicação é feita através de um wsURL (pois o mediador secundário só se regista no UDDI quando o mediador principal vai abaixo).

Requisitos:

Redundância física (canal 3 da Figura 1).

O mecanismo que assegura que o serviço está disponível é implementado através de provas de vida. O mediador principal vai enviar provas de vida de cinco em cinco segundos para o mediador secundário. Se o mediador secundário não receber a prova em seis segundos, este publica o seu wsName no UDDI. Desta forma, os pedidos do cliente são direcionados para o mediador secundário. Este mediador secundário não vai dar provas de vida nem invocar updateShopHistory() nem updateCart(), pois não há nenhum mediador terceário.

O updateShopHistory() adiciona a ShoppingResultView à lista de compras do mediador secundário a shopResultView criada pelo mediador principal.
O updateCart() adiciona a vista à lista de CartViews do mediador secundário a lista criada pelo mediador principal. Quando o addToCart é chamado há duas hipóteses: o item já existe no carro de compras, e só tem de se incrementar o seu valor ou o item não existe e tem de se adicionar ao carrinho. Esta função chama o updateCart(). O updateCart() vai adicionar o CartView correspondente ao mediador secundário.

Redundância temporal (canal 1/2 da Figura 1).

A redundância temporal é obtida através do front-end. O front-end é uma camada que visa abstrair os pedidos do cliente ao mediador. Esta camada lida com problemas de coneção e repetição de pedidos.
Todas as funções do cliente chamam as funções do front-end. O front-end lida com timeouts (na coneção e na resposta). Caso haja um timeout, o cliente procura no UDDI pelo mediador secundário. De seguida, cria os stubs, e repete o pedido.

Redundância informacional (canal 3 da Figura 1).

A redundância de informação é obtida através de um id gerado aleatoriamente, aquando de uma operação buyCart () ou addToCart(). Para isso, criámos um handler MessageIdHandler. Este handler transporta um id gerado para cada operação de addCart() e buyCart(). No mediador, as operações addCart() e buyCart() recebem esse id e guardam-no num mapa. Caso o id recebido já esteja no mapa, significa que a mensagem está repetida e o pedido não deve ser processado de novo.